Final Report for Period: 07/2008 - 06/2009
Submitted on: 07/12/2009
Principal Investigator: Schwan, Karsten .
Award ID: 0702249
Organization: GA Tech Res Corp - GIT
Submitted By:
Gavrilovska, Ada - Co-Principal Investigator
Title:
I/O Virtualization - From Self-Virtualizing Devices to Metadata-Rich Information Appliances

Project Participants

Senior Personnel

Name: Schwan, Karsten
Worked for more than 160 Hours: Yes
Contribution to Project:

Name: Eisenhauer, Greg
Worked for more than 160 Hours: Yes
Contribution to Project:

Name: Gavrilovska, Ada
Worked for more than 160 Hours: Yes
Contribution to Project:

Post-doc

Graduate Student

Name: Kumar, Sanjay
Worked for more than 160 Hours: No
Contribution to Project:

Name: Raj, Himanshu
Worked for more than 160 Hours: No
Contribution to Project:

Name: Kesavan, Mukil
Worked for more than 160 Hours: Yes
Contribution to Project:

Name: Ranadive, Adit
Worked for more than 160 Hours: Yes
Contribution to Project:

Name: Tembey, Priyanka
Worked for more than 160 Hours: No
Contribution to Project:

Name: Gupta, Vishakha
Worked for more than 160 Hours: No
Contribution to Project:

Name: Bhatt, Anish
Worked for more than 160 Hours: No
Contribution to Project:

Undergraduate Student

Name: Kurjanowicz, Matthew
Worked for more than 160 Hours: No
Contribution to Project:

Name: Zakaraia, Omar
Worked for more than 160 Hours: No
Contribution to Project:

Name: Heiskell, Ben
Worked for more than 160 Hours: No
Contribution to Project:

Technician, Programmer

Other Participant

Research Experience for Undergraduates

Organizational Partners

Other Collaborators or Contacts

Interactions with industry researchers at Intel, HP, IBM and VMware.

Activities and Findings

Research and Education Activities: (See PDF version submitted by PI at the end of the report)

Findings: (See PDF version submitted by PI at the end of the report)

Training and Development:

- The project enabled us to support research on the very timely topic of dealing with IO devices and addressing the computational and memory loads IO interactions present in modern manycore virtualized platforms.

- It created opportunities for establishing local knowledge, skillsets and infrastructure to assign class projects based on IO virtualization. These have included several smaller group projects exposing several groups of 2-4 students to skills such as the Xen virtualization platform or modern devices such as smart network adapters or programmable graphics cards. In addition, we were able to assign a related project in the required operating systems taken by all CS graduate students, thereby allowing us to supplement their theoretical understanding of the problems with a strong practical component.

Outreach Activities:

Students and research have benefitted substantially from the project. Three of our students were able to attain summer intern positions at VMWare, for instance. Two summer interns went to IBM, also working on virtualized systems. Two students attained full-time positions at Intel and Microsoft, again working in the virtualization domain. Further, there have been substantial and meaningful research interactions with several of the key companies working in this domain, including IBM, Intel, HP, Microsoft, and VMWare. Finally, all of the code and solutions developed in this project have been open source codes, so that we can freely share them with other institutions. This has led VMWare, for instance, to provide us with an additional award that makes available to our group (and our students) VMWare ESXserver sources. More general outreach efforts have included press releases by Intel Corporation on our efforts on multicore education, and press releases by IBM and HP on our efforts on advancing virtualization technologies to address the broader domain of cloud computing.

Journal Publications

Books or Other One-time Publications

Himanshu Raj, Karsten Schwan,, "'High Performance and Scalable I/O Virtualization via Self-Virtualized Devices'", (2007). conference publication, Published
Bibliography: Proc. of High Performance Distributed Computing (HPDC'07)

Himanshu Raj, Sanjay Kumar, Balasubramanian Seshasayee, Radhika Niranjan, Ada Gavrilovska, Karsten Schwan, "'Enabling Semantic Communications for Virtual Machines via iConnect'", (2007). workshop publication, Published
Bibliography: Workshop on Virtualization Technology in Distributed Computing (VTDC2007), help in conjunction with Supercomputing 2007

Himanshu Raj, Karsten Schwan, "'O2S2: Enhanced Object-based Virtualized Storage'", (2008). workshop publication, Published
Collection: ACM Operating Systems Review, Feb./Oct. 2008.
Bibliography: SPEED'08, in conjunction with HPCA'08

Sanjay Kumar, Karsten Schwan, "'Netchannel: A VMM-level Mechanism for Continuous, Transparent Device Access During VM Migration'", (2008). conference publication, Published
Bibliography: Virtual Execution Environments (VEE'08)

Himanshu Raj, Balasubramanian Seshasayee, Karsten Schwan, "'VMedia: Enhanced Multimedia Services in Virtualized Systems'", (2008). conference publication, Published
Bibliography: Multimedia Computing and Networking (MMCN)

Priyanka Tembey, Anish Bhatt, Dulloor Rao, Ada Gavrilovska, Karsten Schwan, "'Flexible Classification on Heterogeneous Multicore Appliance Platforms'", (2008). conference publication, Accepted
Bibliography: International Conference on Computers, Communication and Networks (ICCCN'08)

Adit Ranadive, Mukil Kesavan, Ada Gavrilovska, Karsten Schwan, "'Performance Implications of Virtualizing Multicore Cluster Machines'", (2008). workshop publication, Published
Bibliography: Workshop on HPC System Virtualization, in conjunction with Eurosys'08

Mukil Kesavan, Adit Ranadive, Ada Gavrilovska, Karsten Schwan, "'Active CoordinaTion (ACT) - Toward Effectively Managing Virtualized Multicore Clouds'", (2008). conference publication, Accepted
Bibliography: Cluster Computing (CLUSTER'08)

Adit Ranadive, Ada Gavrilovska, Karsten Schwan, "IBMon: Monitoring VMM-Bypass Capable InfiniBand Devices using Memory Introspection", (2009). workshop publication, Published
Bibliography: 3rd Workshop on HPC System Virtualization, in conjunction with Eurosys'09

Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia, Vanish Talwar and Partha Ranganathan, "GViM:GPU-accelerated Virtual Machines", (2009). workshop publication, Published
Bibliography: 3rd Workshop on HPC System Virtualization, in conjunction with Eurosys'09

Ada Gavrilovska, Priyanka Tembey, Anish Bhatt, Venkatraghavan Srinivasan, Karsten Schwan, "Supporting Distributed Cloud Services with Heterogeneous Multicore Platforms", (2009). abstract and presentation, Published
Bibliography: presentation at the Intel Embedded and Communications Group Summit

Web/Internet Site

Other Specific Products

Product Type:

Software (or netware)

Product Description:

IB virtualization solution for Xen hypervisor with scalable support for many VMs. The currently available open source solution supports only a single VM, and even that with rather limited capabilities on the control channel.

Sharing Information:

We will contribute this is a patch to the open source Xen distribution.

Product Type:

Software (or netware)

Product Description:

GVim: Virtualization solution for NVIDIA GPU cards for Xen hypervisor. GVim results in very low overheads, as it eliminates all copies from the data movement path; allows virtualization of multi-GPU platforms; and provides fair sharing of GPU resources and support for per-VM QoS support

Sharing Information:

We will make this available as open source.

Product Type:

Software (or netware)

Product Description:

IBMon: tool for asynchronously monitoring and managing virtualized InfiniBand devices in Xen environments, which relies on memory introspection technigies

Sharing Information:

we will make it available online

Contributions

Contributions within Discipline:

- provided software artifacts for efficient virtualization and sharing of InfiniBand and GPU devices;
- introduced the notion of self-virtualizing devices and the device extensions (i.e., logical devices) made possible with SV-IO
- developed and demonstrate the utility of methods for efficient management of virtualized I/O resources
- devloped a tool for asynchronous monitoring of InfiniBand resources, and demonstrated the feasibility of monitoring VMM-bypass devices

Contributions to Other Disciplines:

- scientific applications running in HPC environments heavily utilize both InfiniBand fabrics and GPUs (and other) accelerators; our work demonstrates the feasibility to efficiently virtualize and manage these types of resources.

Contributions to Human Resource Development:

Contributions to Resources for Research and Education:

- we have integrated class projects based on this award into the curriculum and exposed graduate and undergraduate students to issues related to virtualization in general, as well as to specific I/O virtualization issues.

Contributions Beyond Science and Engineering:

Conference Proceedings

Categories for which nothing is reported:

Organizational Partners
Any Journal
Any Web/Internet Site
Contributions: To Any Human Resource Development
Contributions: To Any Beyond Science and Engineering
Any Conference

Findings

In the context of the main components described in the Activities section, we report the following findings:

SV-IO

Experimental evaluations show that both host- and device-resident realizations of the SV-IO abstraction are feasible, and that the self-virtualization concept can deliver significant performance benefits. Device-resident realizations attain negligible virtualization overheads and permit shared, concurrent device access by muliple applications and/or VMs with near native performance. In addition, the approach improves device scalability to support multiple concurrent virtual device interfaces. Detailed results appear in [HPDC07,HPCVirt08]. While the host-resident SV-IO realization does not eliminate the CPU loads required to implement the device sharing mechanisms and policy, the ability to designate host resources (i.e., cores) to this type of functionality eliminates significant host overheads due to reduced context switches, interrupt processing, cache polution and other elements of OS noise [HPDC07].

Logical Devices

Logical devices are a powerful mechanism for presenting applications and guest VMs with device interfaces with richer or novel useful semantics compared to the physical devices available on the platforms. We demonstrate that such functionality is particularly needed to support on a single platform applications/VMs with varied resource requirements or service-level agreements (SLAs), by associating with the virtual device interface corresponding QoS parameters, realized through custom buffer queues, singaling or polling intervals, etc., as reported in [VTDC07,HPCVirt08].

Furthermore, logical devices are a powerful mechanism to present to applications and VMs the device interface they require, even in the absense of such physical device. For instance, we have developed a logical extension of a network device that transparently remotes a physical disk device, thereby ensuring continuous operations of VMs in the event of their migration to nodes where the original disk device is otherwise inacessible. Attained results for a particular workload indicate near 40% reduction in VM downtime compared to other solution [VEE08]. Another demonstration is for logical multimedia devices, where a single physical camera device is shared by multiple processes, each using different camera resolutions, additional functions like image cropping, etc. The utility and performance of solutions like these are reported in [MMCN08].

Finally, logical devices enable selective access to a potentially distributed set of remote devices, each of which with different performance, reliability or other properties. We demonstrate the utility of this by logical device interface corresponding to object-based storage device, realized on top of a regular NIC device. The logical device allows a VM executing on a mobile diskless platform to access a set of distributed and highly diverse storage devices transparently, where the exact physical device accessed is selected based on runtime loads and application performance, reliability and trust requirements [MMCS08].

Smart IO

The ability to associate computation with virtualized device interfaces opens opportunities to modify and extend the functionality otherwise supported by the target physical device with novel types of application-specific processing. The aforementioned QoS, trust and reliability attributes that can be associated with device accesses are some examples of smart IO functionality enabled by our virtualized IO approach [MMCS08,VTDC07,VEE08]. Additional examples include the migration of application-specific data manipulation operations, such as content-based filtering, data reformatting and distribution. We evaluate the utility of smart IO operations for programmable network interface cards based on the IXP network processors, and demonstrate various levels of performance and scalability improvements, including a 5.5 times increase in the throughput levels sustained with a smart network IO device performing content based dispatching of data across a large number of host-resident VMs, as reported in [ICCCN08].

In addition to evaluating the performance benefits of smart IO, we investige the runtime support necessary for efficient execution of application-specific functionality with IO operations. Towards this

end, we have developed a flexible classification mechanism that allows us to tune the classification overhead to the requirements of the types of application processing, or to tune the degree of consistency and synchronization enforced across the host and physical device platforms, again with the objective of controlling the runtime overheads associated with these types of operations [ICCCN08].

Managment of Virtualized IO

One of the key attributes of self-virtualized IO devices is that device resource management is encapsulated in the SV-IO abstraction, and therefore, may be realized on separate device-resident resources or on designated host cores. Through experimentation, we demonstrate that a lack of coordination across SV-IO resource managers and other resource managers present in a virtualized system (e.g., the vCPU scheduler) can result in significant violations of application/VM-expected service levels and SLAs. Namely, in spite of the ability to configure the host (e.g., Xen hypervisor) scheduler to allocate appropriate CPU resources to invidual guest VMs, the inability to translate these requirements to the device (e.g., Infiniband self-virtualized network adapters) can result in 'random' performance levels [HPCVirt08].

In order to address this problem, we have developed a resource management mechanism based on 'active coordination' between multiple resource allocators, and we were able to adapt to dynamic variations in application CPU and IO requirements without any form of static overprovisioning. The resulting approach enabled us to meet application SLAs while at the same time requiring substantially lower physical resources (e.g., up to 50% less CPU and 63% network IO for a mix of workloads consisting of industry-standard benchmarks including RuBiS, Hadoop, iperf and SPEC). Detailed results of this work will be reported in [CLUSTER08].

One key challenge we observed in our work, is that once a VM is provided with direct access to device resources, as is the case for fully self-virtualizable devices, the virtualization layer, i.e., the hypervisor, has no way to further control and manage the VM-device interactions. There may be limited type of device-level monitoring and/or management features providing information on per virtual interface basis (e.g., length of request queues associated with a virtual interface used by a VM), or some information may be available through external devices (e.g., using Subnet Management functionality embedded in InfiniBand routers). However, these solutions are not generally adequate, since (i) current devices (including InfiniBand HCAs) provide only aggregate resource usage information, and not on per VM basis, (ii) any future per VM performance counters will be limited in number and may not be sufficient to support the required management actions, and (iii) these operations may incurr statically configured and prohibitively long latencies. Towards this end, and specifically targeting InfiniBand devices, we developed IBMon, an asynchronous tool which uses VM memory introspection to infer information regarding VM-device interactions, including number and types of accesses, bytes transferred, communication patterns, even specific data being accessed by the device. In [HPCVirt09a] we describe the implementation and the evaluation analysis of the current version of IBMon and demonstrate that it permits monitoring self-virtualized IB devices with acceptable overheads and accuracy, which can be furthermore tuned based on the current needs or operating conditions.

Additional Devices

As part of this project, we were able to successfully realize the virtualization of a novel PCI-Express based IXP-based platform (our prior work used a PCI-based device based on the earlier generation of these processors), and of a PCI-Express based NVIDIA graphics processor.

Regarding the former, with our solution we were able to attain VM data rates close to the 4Gbps limit supported by the physical device without any detectable overheads due to virtualization, and in addition to integrate device-resident SmartIO capabilities to provide various types of VM-specific functionality, as well as dynamically configurable quality of service. These results are partially available in [ICCCN08], and more recently presented at Intel's Embedded and Communication Group Summit [ECG09].

Regarding GPU virtualization, we developed GVim, an efficient solution for virtualizing 'closed' graphics accelerators, i.e., with prorietary software stacks, as is the case with the NVIDIA GPUs targegeted by GVim. Key controbutions of GVim include (i) the ability to fairly share GPU resource across

VMs, including to provide differentiated QoS on per VM basis, (ii) an efficient data movement solution between device and VM memory, which bypasses the need for hypervisor intervention and extra copies, and (iii) demonstration with a range of applications from the HPC and enterprise domain. GVim is described in [HPCVirt09b], and additional publications for both of these efforts are under submission/preparation.

Activities

The focus of this research has been the creation of novel system- and device-level virtualization technology to help address the gap between application level data exchanges and the I/O services provided by underlying hardware and system platforms.

This research has the following main components:

- **SV-IO:** The main ideas behind the virtualized IO devices targeted by this work are captured in the *self-virtualizable* IO devices introduced by our group in [HPDC07]. At minimum, an SV-IO device exports to OSs and applications basic interfaces intended for data movement and invocation of the main operations supported by the underlying physical device. In addition, it enables the efficient sharing of the underlying physical device by multiple concurrent entities (virtual machines and guest OSs or applications), by performing the multiplexing/demultiplexing of device access operations. The actual realization of an SV-IO may require use of designated general purpose resources (i.e., host cores), or it may be implemented on the device itself. The latter option is feasible for open/programmable devices or devices with sufficient resources, such as the IXP network processor-based NICs or the STI Cell accelerators used in our work, or the Infiniband network adapters with hardware support for self-virtualization, with which we have also experimented. The host-based realization of devices is necessary for devices with lower on-device resources (such as some smaller FPGAs) or with closed/proprietary software stacks (such as the CUDA interface for the NVIDIA GPUs used in our research).

- **Logical devices:** In order to address both the semantic gap between application-level data exchanges and the IO services provided by underlying hardware, as well as the capability mismatch between physical IO devices and the processing capacity of host-based platforms, we introduce, implement and evaluate *logical devices*. The intended outcomes are improvements in performance derived from: (1) reductions in the actual volumes of data moved, (2) reductions in data buffering and memory usage, in network and disk usage, and similar resources used in the data path, and (3) improved asynchrony between the data fast path and associated control actions. Logical devices extend the basic API exported by virtualized physical devices with additional functionality. For instance, a camera device may export a logical device which crops or otherwise manipulates camera images, which means that it supports additional API calls to pass the required data handler (e.g., cropping function) and its parameters. Other examples of logical devices targeted by our work include those that support transparent access to remote devices, including to sets of distributed devices, or that export additional QoS levels otherwise not available on the physical device, as reported in the following research contributions resulting from this research [SPEED08,VEE08,VTDC07].

- **Smart IO:** The ability to modify the application or guest VM interface to a physical device creates oportunities for extending the functionality of the physical device with novel application- or VM-specific operations. Examples include data filtering and reformating, the execution of various content based data manipulations, or device-level support for application- or VM-specific QoS. We have experimented with such ideas and have created several concrete implementations of logical device extensions on the various devices targeted by this proposal. In addition, we have investigated and developed the basic runtime mechanisms necessary to support smart IO extensions, including efficient classification and runtime resource management and reconfiguration, as reported in [ICCCN08,SPEED08,MMCN08].

- **Management of virtualized IO:** Self virtualized IO creates additional challenges in ensuring coordinated management of the agregate platform resources - CPUs and IO, necessary to support adequate QoS levels to a range of concurrently executing applications. We have evaluated the need for such coordination and have developed resource management mechanisms and algorithms that address some of these challenges, reported in recent publications (see [HPCVirt08,CLUSTER08]). In addition, and specifically for InfiniBand self-virtualized devices, we created a monitoring tool, IBMon, which allows asynchronous monitoring, and management, of the VM-IB interactions, not otherwise possible with current IB technology (see [HPCVirt09a]).

- **Support for novel device types:** In addition to the IXP-based NICs with which we have worked for several years now, our work also addresses other types of devices (i.e., accelerators), including the STI

Cell processors, NVIDIA graphics processors, FPGAs, and others. We have developed basic virtualization technologies for such devices in order to understand the manner in which they are accessed and shared by end user applications and guest OSs. In particular, we have developed efficient mechanisms to virtualize 'closed' devices without self-virtualization capabilities, such as the NVIDIA GPUs. For such devices, in the absence of device-level virtualization offload and VMM-bypass capabilities, we designed and evaluated mechanisms which enable VMM-bypass for VM-device zero-copy data movements, while the control path, i.e., device access requests, continue to be managed by a host-based realization of the virtualized GPU (see [HPCVirt09b]).